

BBBBBBBBBBBBBBB      AAAAAAAA      SSSSSSSSSSSS      RRRRRRRRRRRRR      TTTTTTTTTTTTTT      LLL  
BBBBBBBBBBBBBBB      AAAAAAAA      SSSSSSSSSSSS      RRRRRRRRRRRRR      TTTTTTTTTTTTTT      LLL  
BBBBBBBBBBBBBBB      AAAAAAAA      SSSSSSSSSSSS      RRRRRRRRRRRRR      TTTTTTTTTTTTTT      LLL  
BBB      BBB      AAA      AAA      SSS      RRR      RRR      TTT      LLL  
BBB      BBB      AAA      AAA      SSS      RRR      RRR      TTT      LLL  
BBB      BBB      AAA      AAA      SSS      RRR      RRR      TTT      LLL  
BBB      BBB      AAA      AAA      SSS      RRR      RRR      TTT      LLL  
BBB      BBB      AAA      AAA      SSS      RRR      RRR      TTT      LLL  
BBB      BBB      AAA      AAA      SSS      RRR      RRR      TTT      LLL  
BBB      BBB      AAA      AAA      SSS      RRR      RRR      TTT      LLL  
BBBBBBBBBBBBBBB      AAA      AAA      SSSSSSSSS      RRRRRRRRRRRRR      TTT      LLL  
BBBBBBBBBBBBBBB      AAA      AAA      SSSSSSSSS      RRRRRRRRRRRRR      TTT      LLL  
BBBBBBBBBBBBBBB      AAA      AAA      SSSSSSSSS      RRRRRRRRRRRRR      TTT      LLL  
BBB      BBB      AAAAAAAAAAAAAA      SSS      RRR      RRR      TTT      LLL  
BBB      BBB      AAAAAAAAAAAAAA      SSS      RRR      RRR      TTT      LLL  
BBB      BBB      AAAAAAAAAAAAAA      SSS      RRR      RRR      TTT      LLL  
BBB      BBB      AAA      AAA      SSS      RRR      RRR      TTT      LLL  
BBB      BBB      AAA      AAA      SSS      RRR      RRR      TTT      LLL  
BBB      BBB      AAA      AAA      SSS      RRR      RRR      TTT      LLL  
BBBBBBBBBBBBBBB      AAA      AAA      SSSSSSSSSSSS      RRR      RRR      TTT      LLL  
BBBBBBBBBBBBBBB      AAA      AAA      SSSSSSSSSSSS      RRR      RRR      TTT      LLL  
BBBBBBBBBBBBBBB      AAA      AAA      SSSSSSSSSSSS      RRR      RRR      TTT      LLL

```

BBBBBBBBBB  AAAAAAAA  SSSSSSSS  CCCCCCCC  VV  VV  TTTTTTTT  TTTTTTTT  PPPPPPPP
BBBBBBBBBB  AAAAAAAA  SSSSSSSS  CCCCCCCC  VV  VV  TTTTTTTT  TTTTTTTT  PPPPPPPP
BB    BB  AA  AA  SS  CC  VV  VV  TT  TT  PP  PP
BB    BB  AA  AA  SS  CC  VV  VV  TT  TT  PP  PP
BB    BB  AA  AA  SS  CC  VV  VV  TT  TT  PP  PP
BB    BB  AA  AA  SS  CC  VV  VV  TT  TT  PP  PP
BBBBBBBBBB  AA  AA  SSSSSS  CC  VV  VV  TT  TT  PPPPPP
BBBBBBBBBB  AA  AA  SSSSSS  CC  VV  VV  TT  TT  PPPPPP
BB    BB  AAAAAAAA  SS  CC  VV  VV  TT  TT  PP
BB    BB  AAAAAAAA  SS  CC  VV  VV  TT  TT  PP
BB    BB  AA  AA  SS  CC  VV  VV  TT  TT  PP
BB    BB  AA  AA  SS  CC  VV  VV  TT  TT  PP
BBBBBBBBBB  AA  AA  SSSSSSSS  CCCCCCCC  VV  VV  TT  TT  PP
BBBBBBBBBB  AA  AA  SSSSSSSS  CCCCCCCC  VV  VV  TT  TT  PP

```

```
1 0001 0 %TITLE 'BASSCVT_T_P - convert numeric text to packed decimal'
2 0002 0 MODULE BASSCVT_T_P (
3 0003 0   IDENT = '1-010'           ! convert numeric text to packed decimal
4 0004 0   ) =                   ! File: BASCVTTP.B32 Edit: MDL1010
5 0005 1 BEGIN
6 0006 1 ****
7 0007 1 *
8 0008 1 *   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 *   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 *   ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 *
13 0013 1 *   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
14 0014 1 *   ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
15 0015 1 *   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
16 0016 1 *   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
17 0017 1 *   OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
18 0018 1 *   TRANSFERRED.
19 0019 1 *
20 0020 1 *   THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
21 0021 1 *   AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
22 0022 1 *   CORPORATION.
23 0023 1 *
24 0024 1 *   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
25 0025 1 *   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
26 0026 1 *
27 0027 1 *
28 0028 1 ****
29 0029 1 *
30 0030 1 *
31 0031 1 ++
32 0032 1   FACILITY: BASIC Language Support
33 0033 1 *
34 0034 1   ABSTRACT:
35 0035 1 *
36 0036 1     This routine converts numeric text to packed decimal.
37 0037 1 *
38 0038 1   ENVIRONMENT: Runs at any access mode - AST reentrant
39 0039 1 *
40 0040 1   AUTHOR: Pamela L. Levesque, CREATION DATE: 29-Dec-1981
41 0041 1 *
42 0042 1   MODIFIED BY:
43 0043 1 *
44 0044 1     1-001 - Original. PLL 29-Dec-1981
45 0045 1     1-002 - Declare BASS$ Psects. PLL 11-Feb-1982
46 0046 1     1-003 - Incorporate changes based on code review. PLL 11-Feb-1982
47 0047 1     1-004 - Fix bug in calculation of integer and fraction digits. PLL 12-Feb-1982
48 0048 1     1-005 - If the input string is all blanks, return 1 integer zero and
49 0049 1       the appropriate number of fractional zeroes.
50 0050 1       Also incorporated DGP's suggestions. PLL 17-Feb-1982
51 0051 1     1-006 - Fix a bug in the all blanks case. PLL 29-Mar-1982
52 0052 1     1-007 - Look at a flag in the frame to determine rounding or truncating.
53 0053 1       PLL 10-Jun-1982
54 0054 1     1-008 - Remove edit 007 - it's the caller's responsibility to check the
55 0055 1       flag in the frame and pass it as an argument if necessary. PLL 30-Jun-1982
56 0056 1     1-009 - when skip_tabs flag is set, the spanc mask should be set to
57 0057 1       skip_blanks OR skip_tabs, not skip_blanks AND skip_tabs. MDL 30-Jun-1983
```

BASS\$CVT\_T\_P  
1-010

BASS\$CVT\_T\_P - convert numeric text to packed de  
16-Sep-1984 00:17:11  
14-Sep-1984 11:54:49

F 10

VAX-11 Bliss-32 v4.0-742  
[BASRTL.SRC]BASCVTP.B32:1

Page 2  
(1)

: 58 0058 1 | 1-010 - check return status from LIB\$\$ADDP\_R7. MDL 13-Jan-1984  
: 59 0059 1 |--  
: 60 0060 1

```
62      0061 1 %SBTTL 'Declarations'  
63      0062 1  
64      0063 1 | SWITCHES:  
65      0064 1  
66      0065 1  
67      0066 1 | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);  
68      0067 1  
69      0068 1  
70      0069 1 | LINKAGES:  
71      0070 1  
72      0071 1  
73      0072 1 | LINKAGE  
74      0073 1   JSB_R7 = JSB (REGISTER = 4, REGISTER = 5, REGISTER = 6, REGISTER = 7) :  
75      0074 1     PRESERVE (8, 9, 10, 11);  
76      0075 1  
77      0076 1 | TABLE OF CONTENTS:  
78      0077 1  
79      0078 1  
80      0079 1 FORWARD ROUTINE  
81      0080 1   BASS$CVT_T_P;                                ! convert from text to packed  
82      0081 1  
83      0082 1  
84      0083 1 | INCLUDE FILES:  
85      0084 1  
86      0085 1  
87      0086 1 LIBRARY 'RTLSTARLE';                      ! System symbols  
88      0087 1  
89      0088 1 REQUIRE 'RTLIN:RTLPSECT';                  ! Define PSECT declarations macros  
90      0183 1  
91      0184 1  
92      0185 1 | MACROS:  
93      0186 1  
94      0187 1  
95      0188 1 | MACRO  
96      0189 1   ADDP = LIB$$ADDP_R7 %:  
97      0190 1  
98      0191 1  
99      0192 1 | EQUATED SYMBOLS:  
100     0193 1  
101     0194 1   NONE  
102     0195 1  
103     0196 1 | FIELDS:  
104     0197 1  
105     0198 1   NONE  
106     0199 1  
107     0200 1 | PSECTS:  
108     0201 1  
109     0202 1  
110     0203 1 | DECLARE_PSECTS (BAS);  
111     0204 1  
112     0205 1  
113     0206 1 | OWN STORAGE:  
114     0207 1  
115     0208 1   NONE  
116     0209 1  
117     0210 1 | EXTERNAL REFERENCES:  
118     0211 1
```

BASS\$CVT\_T\_P  
1-010

BASS\$CVT\_T\_P - convert numeric text to packed de  
Declarations

H 10

16-Sep-1984 00:17:11  
14-Sep-1984 11:54:49

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASCVTIP.B32;1

Page 4  
(2)

```
: 119      0212 1
: 120      0213 1 EXTERNAL ROUTINE
: 121      0214 1 LIB$$ADDP_R7 : JSB_R7;
: 122      0215 1
: 123      0216 1
```

```
125      0217 1 %SBTTL 'BASS$CVT_T_P - convert numeric text to packed decimal'
126      0218 1 GLOBAL ROUTINE BASS$CVT_T_P (
127          0219 1     TEXT_DSC,
128          0220 1     PACKED_DSC,
129          0221 1     FLAGS
130          0222 1     )
131          0223 1
132          0224 1 ++
133          0225 1     FUNCTIONAL DESCRIPTION:
134          0226 1
135          0227 1     Converts a text string representing a numeric value to a packed
136          0228 1     decimal value. Description of the text is as follows:
137          0229 1     <0 or more spaces, nulls, or tabs>
138          0230 1     <"+-" or nothing>
139          0231 1     <0 or more digits>
140          0232 1     <"." or nothing>
141          0233 1     <0 or more digits>
142          0234 1     <0 or more spaces, nulls, or tabs>
143          0235 1     <end of string>
144          0236 1
145          0237 1     Setting the strip spaces bit in FLAGS causes embedded spaces to be ignored.
146          0238 1     Otherwise blanks are an error.
147          0239 1
148          0240 1     Setting the skip tabs bit in FLAGS causes embedded tab characters to be ignored.
149          0241 1     Otherwise they are an error.
150          0242 1
151          0243 1     The value will be rounded, if necessary, to fit into the number of
152          0244 1     digits specified by the packed decimal descriptor. Truncation may be
153          0245 1     requested by setting the don't round bit in FLAGS.
154          0246 1
155          0247 1     Note that this routine expects the length and scale fields in the
156          0248 1     packed decimal descriptor to be set by the caller.
157          0249 1
158          0250 1
159          0251 1     CALLING SEQUENCE:
160          0252 1
161          0253 1     ret_status.wlc.v = BASS$CVT_T_P (TEXT_DSC.rt.dx1, PACKED_DSC.wp.dsd, FLAGS.rl.v)
162          0254 1
163          0255 1     FORMAL PARAMETERS:
164          0256 1
165          0257 1     TEXT_DSC          address of the input text descriptor
166          0258 1     PACKED_DSC        address of the output decimal descriptor
167          0259 1     [FLAGS]           value of caller's flags
168          0260 1     bit 0 set indicates skip blanks
169          0261 1     bit 3 set indicates don't round value
170          0262 1     bit 4 set indicates skip tabs
171          0263 1
172          0264 1     IMPLICIT INPUTS:
173          0265 1
174          0266 1     NONE
175          0267 1
176          0268 1     IMPLICIT OUTPUTS:
177          0269 1
178          0270 1     NONE
179          0271 1
180          0272 1
181          0273 1     ROUTINE VALUE:
```

```

182      0274 1 | success      text successfully converted to decimal
183      0275 1 | failure     numeric text could not be converted to the
184      0276 1 |             specified decimal descriptor
185      0277 1 |
186      0278 1 | SIDE EFFECTS:
187      0279 1 |             NONE
188      0280 1 |
189      0281 1 |
190      0282 1 | ---
191      0283 1 |
192      0284 2 | BEGIN
193      0285 2 |
194      0286 2 | MAP
195      0287 2 |     TEXT_DSC : REF_BLOCK [8,BYTE],
196      0288 2 |     PACKED_DSC : REF_BLOCK [12,BYTE];
197      0289 2 |
198      0290 2 | BUILTIN
199      0291 2 |     ACTUALCOUNT,
200      0292 2 |     CVTSP,
201      0293 2 |     SPANC;
202      0294 2 |
203      0295 2 | LITERAL
204      0296 2 |     K_FLAGS_ARG = 3,
205      0297 2 |     K_PACKED_ONE = %X'0000001C',
206      0298 2 |     K_PACKED_NEG_ONE = %X'0000001D',
207      0299 2 |     K_MAX_DIGITS = 31,
208      0300 2 |     CHAR_TAB = %X'09' : UNSIGNED (8),
209      0301 2 |     CHAR_SPACE = %X'20' : UNSIGNED (8),
210      0302 2 |     CHAR_NULL = %X'00' : UNSIGNED (8),
211      0303 2 |     CHAR_ZERO = %X'30' : UNSIGNED (8),
212      0304 2 |     CHAR_ONE = %X'31' : UNSIGNED (8),
213      0305 2 |     CHAR_TWO = %X'32' : UNSIGNED (8),
214      0306 2 |     CHAR_THREE = %X'33' : UNSIGNED (8),
215      0307 2 |     CHAR_FOUR = %X'34' : UNSIGNED (8),
216      0308 2 |     CHAR_FIVE = %X'35' : UNSIGNED (8),
217      0309 2 |     CHAR_SIX = %X'36' : UNSIGNED (8),
218      0310 2 |     CHAR_SEVEN = %X'37' : UNSIGNED (8),
219      0311 2 |     CHAR_EIGHT = %X'38' : UNSIGNED (8),
220      0312 2 |     CHAR_NINE = %X'39' : UNSIGNED (8),
221      0313 2 |     MASK_NUMERIC = 1 : UNSIGNED (8),
222      0314 2 |     MASK_BLANKS = 2 : UNSIGNED (8),
223      0315 2 |     MASK_TAB = 4 : UNSIGNED (8),
224      0316 2 |     V_SKIP_TABS = 1^4,
225      0317 2 |     V_SKIP_BLANKS = 1^0,
226      0318 2 |     V_DONT_ROUND = 1^3;
227      0319 2 |
228      0320 2 | BIND
229      0321 2 |
230      0322 2 |     SPANC_TABLE = UPLIT BYTE
231      0323 2 |     (
232      0324 2 |     REP CHAR NULL OF (0),
233      0325 2 |     (MASK_BLANKS),
234      0326 2 |     REP CHAR_TAB - CHAR_NULL - 1 OF (0),
235      0327 2 |     (MASK_TAB),
236      0328 2 |     REP CHAR_SPACE - CHAR_TAB - 1 OF (0),
237      0329 2 |     (MASK_BLANKS),
238      0330 2 |     REP CHAR_ZERO - CHAR_SPACE - 1 OF (0),

```

```

239 0331 2      (MASK_NUMERIC),
240 0332 2      REP CHAR ONE - CHAR_ZERO - 1 OF (0),
241 0333 2      (MASK_NUMERIC),
242 0334 2      REP CHAR TWO - CHAR_ONE - 1 OF (0),
243 0335 2      (MASK_NUMERIC),
244 0336 2      REP CHAR THREE - CHAR_TWO - 1 OF (0),
245 0337 2      (MASK_NUMERIC),
246 0338 2      REP CHAR FOUR - CHAR_THREE - 1 OF (0),
247 0339 2      (MASK_NUMERIC),
248 0340 2      REP CHAR FIVE - CHAR_FOUR - 1 OF (0),
249 0341 2      (MASK_NUMERIC),
250 0342 2      REP CHAR SIX - CHAR_FIVE - 1 OF (0),
251 0343 2      (MASK_NUMERIC),
252 0344 2      REP CHAR SEVEN - CHAR_SIX - 1 OF (0),
253 0345 2      (MASK_NUMERIC),
254 0346 2      REP CHAR EIGHT - CHAR_SEVEN - 1 OF (0),
255 0347 2      (MASK_NUMERIC),
256 0348 2      REP CHAR NINE - CHAR_EIGHT - 1 OF (0),
257 0349 2      (MASK_NUMERIC),
258 0350 2      REP 255 - CHAR_NINE OF (0)
259 0351 2      ) : VECTOR [256, BYTE, UNSIGNED];

260 0352 2
261 0353 2 LOCAL
262 0354 2   USER_FLAGS,           flags from caller or 0
263 0355 2   TEMP_LENGTH : INITIAL (1), length of temp. text
264 0356 2           (init to 1 for sign length)
265 0357 2   TEXT_BUF : VECTOR [32, BYTE], temp. area for input text
266 0358 2   INT_DIGITS,          # int. digits in a segment
267 0359 2           of the input string
268 0360 2   TOTAL_INT_DIGITS : INITIAL (0), # int. digits in input
269 0361 2   ACTUAL_SCALE : INITIAL (0), # fract. digits in a segment
270 0362 2           of the input string
271 0363 2   TOTAL_FRAC_DIGITS : INITIAL (0), # fract. digits in input
272 0364 2   DIFF,                temp. variable
273 0365 2   CURRENT_PTR,        ptr. into input text string
274 0366 2   ROUND_DIGIT : INITIAL (0), digit to round on
275 0367 2   MASK,                temp. mask for SPANC
276 0368 2   START_PTR,          ptr. to 1st char in input string
277 0369 2   DEC_PT_LOC : INITIAL (0), ptr. to dec. pt. in input string
278 0370 2   END_TXT_PTR;        ptr. to end of input string
279 0371 2
280 0372 2 MAP
281 0373 2   CURRENT_PTR : REF VECTOR [,BYTE];
282 0374 2
283 0375 2
284 0376 2   !+
285 0377 2   The overall strategy is to construct an acceptable numeric string for CVTSP
286 0378 2   in a temporary area, TEXT_BUF. "Acceptable" means no embedded spaces, tabs,
287 0379 2   or decimal point, and less than or equal to 31 digits. This routine should
288 0380 2   detect any invalid strings which would cause a reserved operand from CVTSP.
289 0381 2
290 0382 2   IF ACTUALCOUNT () LSS K_FLAGS_ARG
291 0383 2   THEN
292 0384 2   USER_FLAGS = 0
293 0385 2   ELSE
294 0386 2   USER_FLAGS = .FLAGS;           ! store caller's flags, if any
295 0387 2

```

```
296      0388 2      END_TXT_PTR = .TEXT_DSC [DSC$A_POINTER] + .TEXT_DSC [DSC$W_LENGTH];
297      0389 2
298      0390 2
299      0391 2      |+ Skip leading spaces. Skip leading tabs only if bit in flag is set.
300      0392 2      |-|
301      0393 2
302      0394 2      IF (.USER_FLAGS AND V_SKIP_TABS) NEQ 0
303      0395 2      THEN
304      0396 2          MASK = MASK_BLANKS OR MASK_TAB
305      0397 2      ELSE
306      0398 2          MASK = MASK_BLANKS;
307      0399 2      CURRENT_PTR = SPANC (.TEXT_DSC [DSC$W_LENGTH], .TEXT_DSC [DSC$A_POINTER],
308      0400 2                  SPANC_TABLE, MASK);
309      0401 2
310      0402 2
311      0403 2      |+ Always insert a sign into the temporary area so that we know a sign
312      0404 2      is always included in TEMP_LENGTH.
313      0405 2      |-|
314      0406 2
315      0407 2      IF .CURRENT_PTR NEQ 0 AND
316      0408 2          (.CURRENT_PTR [0] EQL %C'+' OR
317      0409 2              .CURRENT_PTR [0] EQL %C'-' )
318      0410 2      THEN
319      0411 2          BEGIN
320      0412 2              TEXT_BUF [0] = .CURRENT_PTR [0];
321      0413 2              CURRENT_PTR = .CURRENT_PTR + 1;
322      0414 2          END
323      0415 2      ELSE
324      0416 2          BEGIN
325      0417 2              TEXT_BUF [0] = %C'+';
326      0418 2          END;
327      0419 2
328      0420 2      START_PTR = .CURRENT_PTR;           ! start = 1st possible digit
329      0421 2
330      0422 2
331      0423 2
332      0424 2      |+ Search for integer digits. Embedded tabs and spaces may be ignored, depending
333      0425 2      on USER_FLAGS.
334      0426 2      |-|
335      0427 2
336      0428 2      WHILE 1 DO
337      0429 2      BEGIN
338      0430 2
339      0431 2
340      0432 2      |+ Wait until here to check for all blanks in the input string.
341      0433 2      A zero string must contain the proper number of fractional digits,
342      0434 2      so it's easiest to let zeroes be handled the same as non-zeroes.
343      0435 2      |-|
344      0436 2
345      0437 2      IF .CURRENT_PTR EQL 0
346      0438 2      THEN
347      0439 2          BEGIN
348      0440 2              TOTAL_INT_DIGITS = 1;
349      0441 2              TEXT_BUF [1] = %C'0';
350      0442 2              TEMP_LENGTH = .TEMP_LENGTH + 1;
351      0443 2              EXIT[00P
352      0444 2          END;
```

```
353 0445 3
354 0446 3 CURRENT_PTR = SPANC (%REF (.END_TXT_PTR - .START_PTR),
355 0447 3 .CURRENT_PTR, SPANC_TABLE, %REF (MASK_NUMERIC));
356 0448 3
357 0449 3 !+ Store any digits found.
358 0450 3
359 0451 3
360 0452 3
361 0453 4 INT_DIGITS = (IF .CURRENT_PTR NEQ 0
362 0454 4 THEN .CURRENT_PTR - .START_PTR
363 0455 3 ELSE .END_TXT_PTR - .START_PTR);
364 0456 3 ! calc # integer digits
365 0457 3 IF .INT_DIGITS GTR 0
366 0458 3 THEN
367 0459 4 BEGIN
368 0460 4 CH$MOVE (.INT_DIGITS, .START_PTR, TEXT_BUF + .TEMP_LENGTH);
369 0461 4 ! copy int. digits to temp area
370 0462 4 TEMP_LENGTH = .TEMP_LENGTH + .INT_DIGITS;
371 0463 4 TOTAL_INT_DIGITS = .TOTAL_INT_DIGITS + .INT_DIGITS;
372 0464 3 END;
373 0465 3
374 0466 3 IF .CURRENT_PTR NEQ 0
375 0467 3 THEN
376 0468 4 BEGIN
377 0469 4 !+ Have found a non-digit. It could be a decimal point or embedded blanks, etc.
378 0470 4
379 0471 4
380 0472 4 SELECTONE .CURRENT_PTR [0] OF
381 0473 4 SET
382 0474 4 [%C'.']:
383 0475 5 BEGIN
384 0476 5 DEC_PT_LOC = .CURRENT_PTR; ! save location of dec pt
385 0477 5 CURRENT_PTR = .CURRENT_PTR + 1;
386 0478 5 EXITLOOP
387 0479 4 END;
388 0480 4 [CHAR_SPACE, CHAR_NULL]:
389 0481 5 BEGIN
390 0482 5 IF (.USER_FLAGS AND V_SKIP_BLANKS) NEQ 0
391 0483 5 THEN
392 0484 6 BEGIN
393 0485 6 CURRENT_PTR = SPANC (%REF (.END_TXT_PTR - .START_PTR),
394 0486 6 .CURRENT_PTR, SPANC_TABLE,
395 0487 6 %REF (MASK_BLANKS));
396 0488 6 IF .CURRENT_PTR EQL 0 THEN EXITLOOP;
397 0489 6 START_PTR = .CURRENT_PTR;
398 0490 6 END
399 0491 5 ELSE
400 0492 5 RETURN 0; ! <should be zeroes here>
401 0493 4 END;
402 0494 4 [CHAR_TAB]:
403 0495 5 BEGIN
404 0496 5 IF (.USER_FLAGS AND V_SKIP_TABS) NEQ 0
405 0497 5 THEN
406 0498 6 BEGIN
407 0499 6 CURRENT_PTR = SPANC (%REF (.END_TXT_PTR - .START_PTR),
408 0500 6 .CURRENT_PTR, SPANC_TABLE,
409 0501 6 %REF (MASK_TAB));
```

```
410      0502 6           IF .CURRENT_PTR EQ 0 THEN EXITLOOP;
411      0503 6           START_PTR = .CURRENT_PTR;
412      0504 6           END
413      0505 5           ELSE
414      0506 5           RETURN 0;
415      0507 4           END;
416      0508 4           [OTHERWISE]:
417      0509 4           RETURN 0;           ! no other chars valid here
418      0510 4           TES;
419      0511 4           END
420      0512 3           ELSE
421      0513 3
422      0514 3
423      0515 3           Only digits found. Purely integer number.
424      0516 3
425      0517 3           EXITLOOP;
426      0518 3
427      0519 2           END;           ! end of WHILE loop
428      0520 2
429      0521 2
430      0522 2           Return an error if the number of integer digits exceeds what the destination
431      0523 2           descriptor specifies.
432      0524 2
433      0525 2
434      0526 3           IF .TOTAL_INT_DIGITS GTR (.PACKED_DSC [DSC$W_LENGTH] +
435      0527 3           .PACKED_DSC [DSC$B_SCALE])
436      0528 2           THEN RETURN 0;
437      0529 2
438      0530 2
439      0531 2           Search for fractional digits if a decimal point was found and we are not
440      0532 2           at the end of the string.
441      0533 2
442      0534 2
443      0535 2           IF .DEC_PT_LOC NEQ 0 AND
444      0536 2           .CURRENT_PTR NEQ 0 AND
445      0537 2           .CURRENT_PTR LSS .END_TXT_PTR
446      0538 2           THEN
447      0539 3           BEGIN
448      0540 3           START_PTR = .DEC_PT_LOC + 1;
449      0541 3
450      0542 3           WHILE 1 DO
451      0543 3           BEGIN
452      0544 4           CURRENT_PTR = SPANC (%REF (.END_TXT_PTR - .START_PTR),
453      0545 4           .CURRENT_PTR, SPANC_TABLE, %REF ?MASK_NUMERIC));
454      0546 4
455      0547 4
456      0548 4
457      0549 4           Copy any digits found into temp. area.
458      0550 4
459      0551 4
460      0552 5           ACTUAL_SCALE = (IF .CURRENT_PTR NEQ 0
461      0553 5           THEN .CURRENT_PTR - .START_PTR
462      0554 4           ELSE .END_TXT_PTR - .START_PTR);
463      0555 4           ! calc # of fract digits
464      0556 4           IF .ACTUAL_SCALE NEQ 0
465      0557 4           THEN
466      0558 5           BEGIN
```

467 0559 5 CH\$MOVE (.ACTUAL\_SCALE, .START\_PTR, TEXT\_BUF + .TEMP\_LENGTH);  
468 0560 5 ! copy fract to temp area  
469 0561 5 TEMP\_LENGTH = .TEMP\_LENGTH + .ACTUAL\_SCALE;  
470 0562 5 TOTAL\_FRAC\_DIGITS = .TOTAL\_FRAC\_DIGITS + .ACTUAL\_SCALE;  
471 0563 4 END;  
472 0564 4  
473 0565 4 IF .CURRENT\_PTR NEQ 0 AND  
474 0566 4 .CURRENT\_PTR LSS .END\_TXT\_PTR  
475 0567 4 ! at end of string?  
476 0568 4 THEN  
477 0569 5 BEGIN  
478 0570 5 !+  
479 0571 5 Have found a non-digit. It may be an embedded blank or tab, or trailing  
480 0572 5 blank or tab.  
481 0573 5 !-  
482 0574 5 SELECTONE .CURRENT\_PTR [0] OF  
483 0575 5 SET  
484 0576 5 [CHAR\_SPACE, CHAR\_NULL]:  
485 0577 6 BEGIN  
486 0578 6 IF (.USER\_FLAGS AND V\_SKIP\_BLANKS) NEQ 0  
487 0579 6 THEN  
488 0580 7 BEGIN  
489 0581 7 CURRENT\_PTR = SPANC (%REF (.END\_TXT\_PTR - .START\_PTR),  
490 0582 7 CURRENT\_PTR, SPANC\_TABLE,  
491 0583 7 %REF (MASK\_BLANKS));  
492 0584 7 IF .CURRENT\_PTR EQL 0 THEN EXITLOOP;  
493 0585 7 START\_PTR = .CURRENT\_PTR;  
494 0586 7 END  
495 0587 6 ELSE  
496 0588 6 RETURN 0; ! ret an error  
497 0589 5 END;  
498 0590 5 [CHAR\_TAB]:  
499 0591 6 BEGIN  
500 0592 6 IF (.USER\_FLAGS AND V\_SKIP\_TABS) NEQ 0  
501 0593 6 THEN  
502 0594 7 BEGIN  
503 0595 7 CURRENT\_PTR = SPANC (%REF (.END\_TXT\_PTR - .START\_PTR),  
504 0596 7 CURRENT\_PTR, SPANC\_TABLE,  
505 0597 7 %REF (MASK\_TAB));  
506 0598 7 IF .CURRENT\_PTR EQL 0 THEN EXITLOOP;  
507 0599 7 START\_PTR = .CURRENT\_PTR;  
508 0600 7 END  
509 0601 6 ELSE  
510 0602 6 RETURN 0;  
511 0603 5 END;  
512 0604 5 [OTHERWISE]:  
513 0605 5 RETURN 0; ! invalid character  
514 0606 5 TES:  
515 0607 5  
516 0608 5 END  
517 0609 5  
518 0610 4 ELSE  
519 0611 4 !+  
520 0612 4 Fraction digits were the rest of the string.  
521 0613 4 !-  
522 0614 4 EXITLOOP;  
523 0615 4

```
: 524      0616 3      END:                                ! End of 'WHILE' Loop
: 525      0617 3
: 526      0618 2      END:                                ! end of fract. digit search
: 527      0619 2
: 528      0620 2      + Does the input text use all the fractional digits allowed by the decimal
: 529      0621 2      descriptor?
: 530      0622 2
: 531      0623 2
: 532      0624 2      DIFF = (-.PACKED_DSC [DSC$B_SCALE]) - .TOTAL_FRAC_DIGITS;
: 533      0625 2      IF .DIFF GTR 0
: 534      0626 2      THEN
: 535      0627 3      BEGIN
: 536      0628 3      + Fewer fractional digits than allowed. Pad with zeroes.
: 537      0629 3      -  

: 538      0630 3      CH$COPY (.TEMP_LENGTH, TEXT_BUF, %C'0', .TEMP_LENGTH + .DIFF,
: 539      0631 3      TEXT_BUF);
: 540      0632 3      TEMP_LENGTH = .TEMP_LENGTH + .DIFF;
: 541      0633 3      END
: 542      0634 3
: 543      0635 3
: 544      0636 3
: 545      0637 2      ELSE IF .DIFF LSS 0
: 546      0638 2      THEN
: 547      0639 3      BEGIN
: 548      0640 3      + Too many fractional digits. Save the first digit past the allowed number
: 549      0641 3      of fractional digits for rounding. The value will be truncated if the user
: 550      0642 3      set dont_round in USER_FLAGS, or if the digit to round on is less than five.
: 551      0643 3      Note that if the caller passed an input string longer than 31 digits, the
: 552      0644 3      32nd digit is saved for rounding purposes. (It is lost after the CVTSP
: 553      0645 3      instruction.)
: 554      0646 3
: 555      0647 3      !-
: 556      0648 3
: 557      0649 3      LOCAL
: 558      0650 3      INDEX;
: 559      0651 3
: 560      0652 3      INDEX = 1                                ! for sign
: 561      0653 3      + .TOTAL_INT_DIGITS
: 562      0654 3      + .TOTAL_FRAC_DIGITS
: 563      0655 3      - 1;                                start counting from 0
: 564      0656 3      ROUND_DIGIT = .TEXT_BUF [.INDEX]; save digit to round on
: 565      0657 3
: 566      0658 3      TEMP_LENGTH = .TEMP_LENGTH + .DIFF; subtr extra fract digits
: 567      0659 3
: 568      0660 3      ! from text string
: 569      0661 2      END;
: 570      0662 2
: 571      0663 2      + Copy the text to the destination descriptor and convert to packed at the
: 572      0664 2      same time.
: 573      0665 2
: 574      0666 2
: 575      0667 2
: 576      0668 2      TEMP_LENGTH = MIN (.TEMP_LENGTH - 1, K_MAX_DIGITS);
: 577      0669 2      ! not more than 31 digits and
: 578      0670 2      ! don't count sign
: 579      0671 2      CVTSP (TEMP_LENGTH, TEXT_BUF, PACKED_DSC [DSC$W_LENGTH],
: 580      0672 2      .PACKED_DSC [DSC$A_POINTER]);
```

```

581      0673 2
582      0674 2
583      0675 2 + see if we need to round this number. the number will be rounded if the
584      0676 2 | user did a SET ROUND and there is a rounding digit >= 5.
585      0677 2 -
586      0678 2 | IF (.USER_FLAGS AND V_DONT_ROUND) EQL 0 AND
587      0679 2 | .ROUND_DIGIT GEQ %C'5'
588      0680 2 | THEN
589      0681 3 | BEGIN
590      0682 3 | LOCAL
591      0683 3 | ONE:
592      0684 3 | IF .TEXT_BUF [0] EQL %C'-
593      0685 3 | THEN
594      0686 3 | ONE = K_PACKED_NEG_ONE
595      0687 3 | ELSE
596      0688 3 | ONE = K_PACKED_ONE;
597      0689 3 |+
598      0690 3 | add one to the last significant digit in the number to cause
599      0691 3 | the rounding effect.
600      0692 3 |-
601      0693 3 | IF NOT ADDP (1, ONE, .PACKED_DSC [DSC$W_LENGTH], .PACKED_DSC [DSC$A_POINTER])
602      0694 3 | THEN RETURN 0;
603      0695 2 | END;
604      0696 2
605      0697 2 | RETURN 1;
606      0698 2
607      0699 1 | END:

```

! End of routine BASS\$CVT\_T\_P

```

.TITLE BASS$CVT_T_P BASS$CVT_T_P - convert numeric text
          to packed de
.IDENT \1-010\
.PSECT _BASS$CODE,NOWRT, SHR, PIC,2

```

02 00000 P.AAA:	.BYTE 2
00# 00001	.BYTE 0[8]
04 00009	.BYTE 4
00# 0000A	.BYTE 0[22]
02 00020	.BYTE 2
00# 00021	.BYTE 0[15]
01 00030	.BYTE 1
01 00031	.BYTE 1
01 00032	.BYTE 1
01 00033	.BYTE 1
01 00034	.BYTE 1
01 00035	.BYTE 1
01 00036	.BYTE 1
01 00037	.BYTE 1
01 00038	.BYTE 1
01 00039	.BYTE 1
00# 0003A	.BYTE 0[198]

SPANC\_TABLE= P.AAA  
.EXTRN LIB\$\$ADDP\_R7

OFFC 00000

.ENTRY BASS\$CVT\_T\_P, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 0218



04	AE	06	12	000AD	BNEQ	14\$	0476	
		89	9E	000AF	MOVAB	(CURRENT_PTR)+, DEC_PT_LOC	0475	
		36	11	000B3	BRB	21\$	0480	
		69	95	000B5	14\$:	TSTB (CURRENT_PTR)		
		05	13	000B7	BEQL	15\$		
20		69	91	000B9	CMPB	(CURRENT_PTR), #32		
		11	12	000BC	BNEQ	18\$		
03		6E	E8	000BE	BLBS	USER_FLAGS, 17\$	0482	
		0142	31	000C1	16\$:	BRW 43\$		
		5A	2B	000C4	17\$:	R10, (CURRENT_PTR), SPANC_TABLE, #2	0487	
		14	13	000CB	BEQL	19\$		
		14	11	000CD	BRB	20\$		
09		69	91	000CF	18\$:	CMPB (CURRENT_PTR), #9	0485	
		ED	12	000D2	BNEQ	16\$	0494	
		69	E9	000D4	BLBC	28(SP), 16\$	0496	
04	FE36	CF	5A	2B	000D8	SPANC	R10, (CURRENT_PTR), SPANC_TABLE, #4	0501
		02	12	000DF	BNEQ	20\$		
		51	D4	000E1	19\$:	CLRL R1		
		59	51	D0	000E3	MOVL R1, CURRENT_PTR	0499	
		03	13	000E6	BEQL 21\$	21\$	0502	
		FF76	31	000E8	BRW 8\$			
		58	08	AC	D0	PACKED_DSC, R8	0526	
		50	68	3C	000EF	MOVZWL (R8), R0	0527	
		51	08	A8	98	CVTBL 8(R8), R1		
		50	51	C0	000F6	ADDL2 R1, R0		
		50	14	AE	D1	TOTAL_INT_DIGITS, R0	0526	
			C2	14	000FD	BGTR 16\$		
		04	AE	D5	000FF	TSTL DEC_PT_LOC	0535	
			02	13	00102	BEQL 22\$		
			59	D5	00104	TSTL CURRENT_PTR	0536	
			7D	13	00106	BEQL 34\$		
		57	59	D1	00108	CURRENT_PTR, END_TXT_PTR	0537	
			7F	18	0010B	BEQL 35\$		
		04	AE	01	C1	#1, DEC_PT_LOC, START_PTR	0541	
		58	57	5B	C3	START_PTR, END_TXT_PTR, R10	0545	
		5A	69	5A	2B	R10, (CURRENT_PTR), SPANC_TABLE, #1	0546	
			02	12	0011D	BNEQ 24\$		
			51	D4	0011F	CLRL R1		
		59	51	D0	00121	MOVL R1, CURRENT_PTR	0545	
			18	AE	D4	CLRL 24(SP)	0552	
			59	D5	00127	TSTL CURRENT_PTR		
			0A	13	00129	BEQL 25\$		
10	AE	18	AE	D6	0012B	INCL 24(SP)		
		59	58	C3	0012E	SUBL3 START_PTR, CURRENT_PTR, ACTUAL_SCALE	0553	
			04	11	00133	BRB 26\$		
10	AE	5A	D0	00135	25\$:	MOVL R10, ACTUAL_SCALE	0554	
24	AE46	6B	10	AE	10	BEQL 27\$	0556	
			AE	28	0013B	MOV C3 ACTUAL_SCALE, (START_PTR), TEXT_BUF-[TEMP LENGTH]	0559	
		0C	56	10	AE	ADDL2 ACTUAL_SCALE, TEMP LENGTH	0561	
		AE	10	AE	C0	ADDL2 ACTUAL_SCALE, TOTAL_FRAC_DIGITS	0562	
		3D	18	AE	E9	BLBC 24(SP), 35\$	0565	
		57	59	D1	0014F	CMPL CURRENT_PTR, END_TXT_PTR	0566	
			38	18	00152	BGEQ 35\$		
			69	95	00154	TSTB (CURRENT_PTR)		
			05	13	00156	BEQL 28\$		
		20	69	91	00158	CMPB (CURRENT_PTR), #32	0576	

BASSCVT\_T\_P  
1-010

BAS\$CVT\_T\_P - convert numeric text to packed de 16-Sep-1984 00:17:11 VAX-11 Bliss-32 V4.0-742  
BAS\$CVT\_T\_P - convert numeric text to packed de 14-Sep-1984 11:54:49 [BASRTL.SRC]BASCVTTP.B32;1

6 11

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASCVTTP.B32;1

Page 16  
(3)

02	FD97	CF	03	11	12	0015B	00A3	11	12	0015D	28\$:	BNEQ	31\$	0578
			69	6E	E8	0015D		31	31	00160	29\$:	BLBS	USER_FLAGS, 30\$	
				5A	2B	00163		14	13	0016A	30\$:	BRW	43\$	
								14	11	0016C		SPANC	R10, (CURRENT_PTR), SPANC_TABLE, #2	0583
			09	69	91	0016E		ED	12	00171	31\$:	BEQL	32\$	0581
												BRB	33\$	0590
04	FD83	CF	E9	1C	AE	E9	00173					CMPB	(CURRENT_PTR), #9	
			69	5A	2B	00177					BNEQ	29\$		
				02	12	0017E					BLBC	28(SP), 29\$		
								51	D4	00180	32\$:	SPANC	R10, (CURRENT_PTR), SPANC_TABLE, #4	0592
			59	51	D0	00182					BEQL	33\$	0597	
				05	13	00185					MOVL	R1, CURRENT_PTR	0595	
			5B	59	D0	00187					BEQL	35\$	0598	
				86	11	0018A					MOVL	CURRENT_PTR, START_PTR	0599	
			57	86	A8	98	0018C	08	A8	98	35\$:	BRB	23\$	0592
				57	0C	AE	00190					CVTBL	8(R8), R7	0624
			57	57	CE	00194					ADDL2	TOTAL_FRAC_DIGITS, R7		
				0E	15	00197					MNEGL	R7, DIFF		
50	50	56	57	57	C1	00199					BLEQ	36\$		
	30	24	AE	56	2C	0019D					ADDL3	DIFF, TEMP_LENGTH, R0	0625	
		24		56	2C	001A3					MOVC5	TEMP_LENGTH, TEXT_BUF, #48, R0, TEXT_BUF	0632	
				24	AE	001A3								
					11	11	001A5					BRB	37\$	0634
					12	18	001A7	36\$:				BGEQ	38\$	0637
	50	51	51	14	AE	D0	001A9					MOVL	TOTAL_INT_DIGITS, R1	0655
		08	51	0C	AE	C1	001AD					ADDL3	TOTAL_FRAC_DIGITS, R1, INDEX	
			AE	24	AE	40	9A	001B2				MOVZBL	TEXT_BUF[INDEX], ROUND_DIGIT	0656
			56	57	C0	001B8		37\$:				ADDL2	DIFF, TEMP_LENGTH	0658
			50	FF	A6	9E	001BB	38\$:				MOVAB	-1(R6), R0	0668
			1F	50	D1	001BF						CMPL	R0, #31	
				03	15	001C2						BLEQ	39\$	
			50	50	1F	D0	001C4					MOVL	#31, R0	
			56	50	D0	001C7	39\$:					MOVL	R0, TEMP_LENGTH	
04	B8	68	24	AE	56	09	001CA					CVTSP	TEMP_LENGTH, TEXT_BUF, (R8), a4(R8)	0672
		2D		6E	03	E0	001D1					BBS	#3, USER_FLAGS, 42\$	0678
			35	08	AE	D1	001D5					CMPL	ROUND_DIGIT, #53	0679
					27	19	001D9					BLSS	42\$	
			2D	24	AE	91	001DB					CMPB	TEXT_BUF, #45	0684
					06	12	001DF					BNEQ	40\$	
			20	AE	1D	D0	001E1					MOVL	#29, ONE	0686
					04	11	001E5					BRB	41\$	
			20	AE	1C	D0	001E7	40\$:				MOVL	#28, ONE	0688
				55	20	AE	9E	001EB	41\$:			MOVAB	ONE, R5	0693
				57	04	A8	D0	001EF				MOVL	4(R8), R7	
				56	68	3C	001F3					MOVZWL	(R8), R6	
				54	01	D0	001F6					MOVL	#1, R4	
				04	0000000G	00	16	001F9				JSB	LIB\$SADDP_R7	
				50	50	E9	001FF					BLBC	R0, 43\$	
				50	01	D0	00202	42\$:				MOVL	#1, R0	0697
												RET		
								50	D4	00206	43\$:	CLRL	R0	
												RET		0699

; Routine Size: 521 bytes, Routine Base: \_BASSCODE + 0100

BASS\$CVT\_T\_P  
1-010

BASS\$CVT\_T\_P - convert numeric text to packed de 16-Sep-1984 00:17:11  
BASS\$CVT\_T\_P - convert numeric text to packed de 14-Sep-1984 11:54:49

H 11

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASCVTTIP.B32;1

Page 17  
(3)

: 608 0700 1  
: 609 0701 1 END  
: 610 0702 1  
: 611 0703 0 ELUDOM

: ! End of module BASS\$CVT\_T\_P

#### PSECT SUMMARY

Name	Bytes	Attributes
_BASS\$CODE	777	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

#### Library Statistics

File	----- Symbols -----			Pages Mapped	Processing Time
	Total	Loaded	Percent		
\$_255\$DUA28:[SYSLIB]STARLET.L32:1	9776	3	0	581	00:01.1

#### COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:BASCVTTIP/OBJ=OBJ\$:BASCVTTIP MSRC\$:BASCVTTIP/UPDATE=(ENH\$:BASCVTTIP)

: Size: 521 code + 256 data bytes  
: Run Time: 00:15.2  
: Elapsed Time: 00:35.7  
: Lines/CPU Min: 2780  
: Lexemes/CPU-Min: 12506  
: Memory Used: 203 pages  
: Compilation Complete

0021 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

BASDET  
LIS

BASDISPAT  
LIS

BASCUTTP  
LIS

BASDELETE  
LIS

BASDATEI  
LIS

BASECHO  
LIS

BASCUTRP  
LIS